Fingerprinting the Checker Policies of Parallel File Systems

Runzhou Han, Duo Zhang, Mai Zheng

IOWA STATE UNIVERSITY

Parallel File Systems (PFSes)

- PFS is the cornerstone of high performance computing
  - Optimized for highly concurrent access

Lustre®

BeeGFS®

IBM Spectrum Scale

OrangeFS™

panasas®

PFS Failures: Real-World Cases

Thursday, September 1, 2016 at 9:10:12 AM Central Daylight Time

Subject: Update: HPCC Power Outage
Date: Monday, January 11, 2016 at 8:50:17 AM Central Standard Time
From: HPCC - Support
Attachments: image001.png, image003.png

TEXAS TECH UNIVERSITY
Information Technology Division

High Performance Computing Center

To All HPCC Customers and Partners,

As we have informed you earlier, the Experimental Sciences Building experienced a major power outage Sunday, Jan. 3 and another set of outages Tuesday, Jan. 5 that occurred while file systems were being recovered from the first outage. As a result, there were major losses of important parts of the file systems for the work, scratch and certain experimental group special Lustre areas.

The HPCC staff have been working continuously since these events on recovery procedures to try to restore as much as possible of the affected file systems. These procedures are extremely time-consuming, taking days to complete in some cases. Although about a third of the affected file systems have been recovered, work continues on this effort and no time estimate is possible at present.

User home areas have been recovered successfully. At present, no user logins are being permitted while recovery efforts proceed on the remaining Lustre areas. Your understanding and patience are appreciated.

If you have questions, please contact us at hpccsupport@ttu.edu or 806-742-4350. Thanks.

Sincerely,
HPCC Staff

Case1: HPCC Power Outage

PFS Failures: Real-World Cases

Thursday, September 1, 2016 at 9:10:12 AM Central Daylight Time

Subject: Update: HPCC Power Outage
Date: Monday, January 11, 2016 at 8:50:17 AM Central Standard Time
From: HPCC - Support
Attachments: image001.png, image003.png

TEXAS TECH UNIVERSITY
Information Technology Division

High Performance Computing Center

To All HPCC Customers and Partners,

As we have informed you earlier, the Experimental Sciences Building experienced a major power outage Sunday, Jan. 3 and another set of outages Tuesday, Jan. 5 that occurred while file systems were being recovered from the first outage. As a result, there were major losses of important parts of the file systems for the work, scratch and certain experimental group special Lustre areas.

The HPCC staff have been working continuously since these events on recovery procedures to try to restore as much as possible of the affected file systems. These procedures are extremely time-consuming, taking days to complete in some cases. Although about a third of the affected file systems have been recovered, work continues on this effort and no time estimate is possible at present.

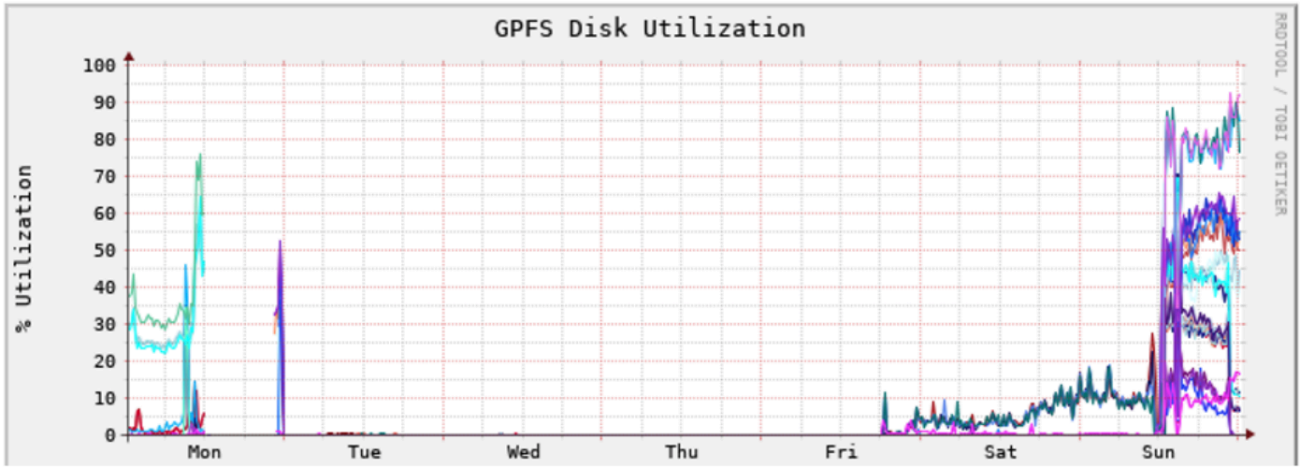
User home areas have been recovered successfully. At present, no user logins are being permitted while recovery efforts proceed on the remaining Lustre areas. Your understanding and patience are appreciated.

If you have questions, please contact us at hpccsupport@ttu.edu or 806-742-4350. Thanks.

Sincerely,
HPCC Staff

Case1: HPCC Power Outage

The graph below offers a vivid example of how bad things got at a prestigious American university[2] which suffered frequent HPC storage outages and took several days to get their systems back up and running. Recovery experienced at this university[3] is shown in the graph below with additional detail available via the link in the footnote. It shows an outage that started on a Monday and wasn't fully recovered from until Sunday.



Case2: ACCRE Storage Outage*

* Hyperion Research survey of HPC organizations done for Panasas

PFS Failures: More Frequent/Expensive Than You Thought

Some statistics*:

The average HPC storage system failure frequency is

9.8 failures/year

≈half

of HPC sites experience storage system failures 1/month or more frequently

* Hyperion Research survey of HPC organizations done for Panasas

PFS Failures: More Frequent/Expensive Than You Thought

Some statistics*:

The average HPC storage system failure frequency is

9.8 failures/year

Downtime ranges from

1 day↓ to 1 week↑

≈half

of HPC sites experience storage system failures 1/month or more frequently

40%

of HPC sites typically took more than 2 weeks to restore their storage systems

* Hyperion Research survey of HPC organizations done for Panasas

PFS Failures: More Frequent/Expensive Than You Thought

Some statistics*:

The average HPC storage system failure frequency is **9.8 failures**/year

Downtime ranges from **1 day↓ to 1 week↑**

A single day of downtime costs from **$100K↓ to $1M↑**

≈half of HPC sites experience storage system failures 1/month or more frequently

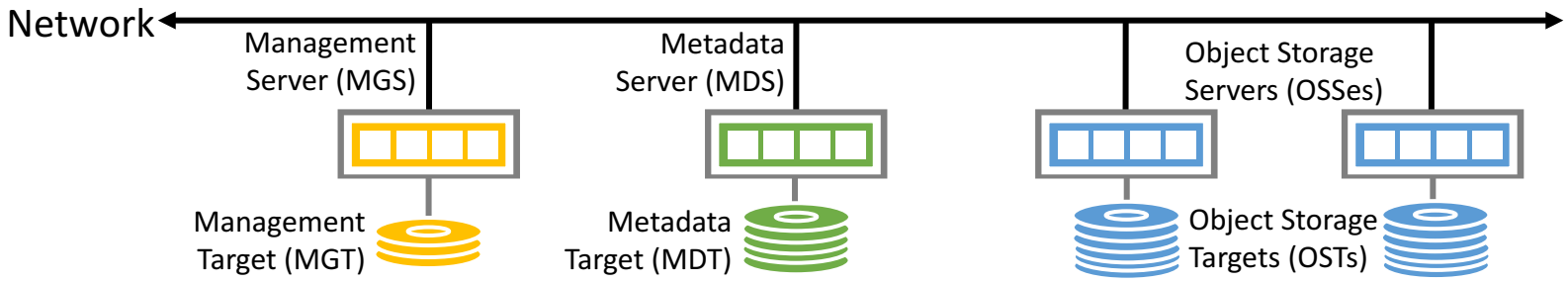
40% of HPC sites typically took more than 2 weeks to restore their storage systems

Average downtime cost is **$127K**/day

* Hyperion Research survey of HPC organizations done for Panasas

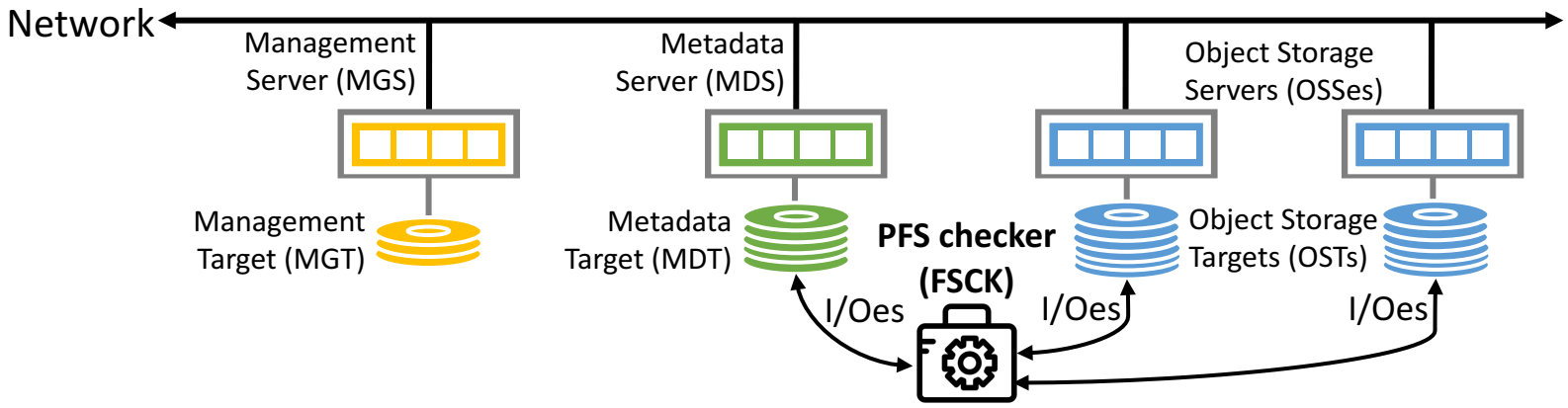
PFS & PFS Checkers (FSCKs)

- Typical PFS architecture



PFS & PFS Checkers (FSCKs)

- Typical PFS architecture

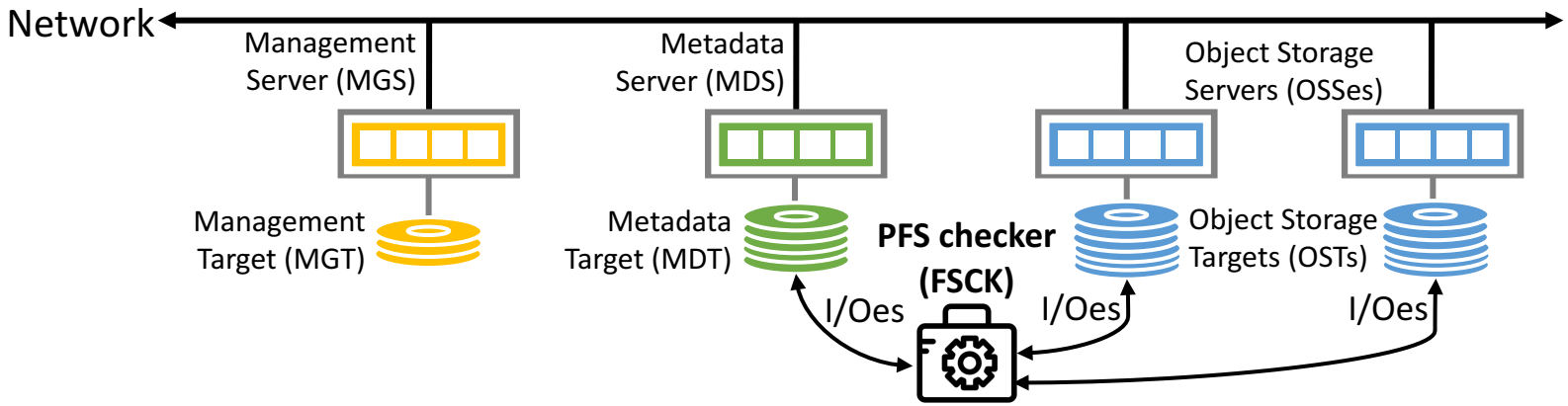


- Many PFSes are designed with a *checker* component
  - e.g., LFSCK for Lustre, BeeGFS-FSCK for BeeGFS, PV2FS-FSCK for OrangeFS

Lustre® BeeGFS® OrangeFS

PFS & PFS Checkers (FSCKs)

- Typical PFS architecture

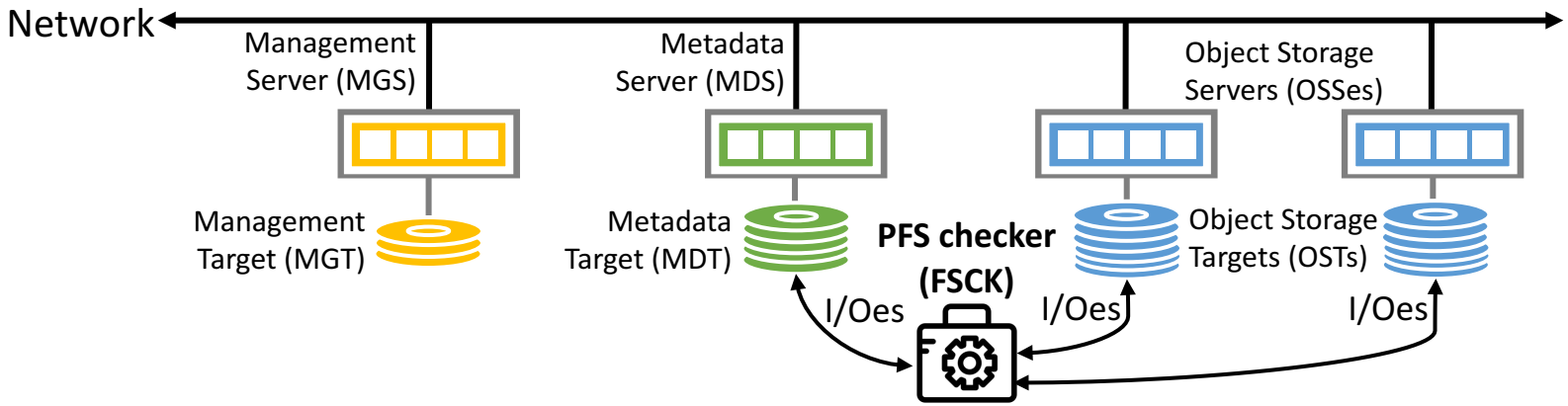


- Many PFSes are designed with a *checker* component
  - e.g., LFSCK for Lustre, BeeGFS-FSCK for BeeGFS, PV2FS-FSCK for OrangeFS
  - Detect and repair inconsistencies

Lustre® BeeGFS® OrangeFS

PFS & PFS Checkers (FSCKs)

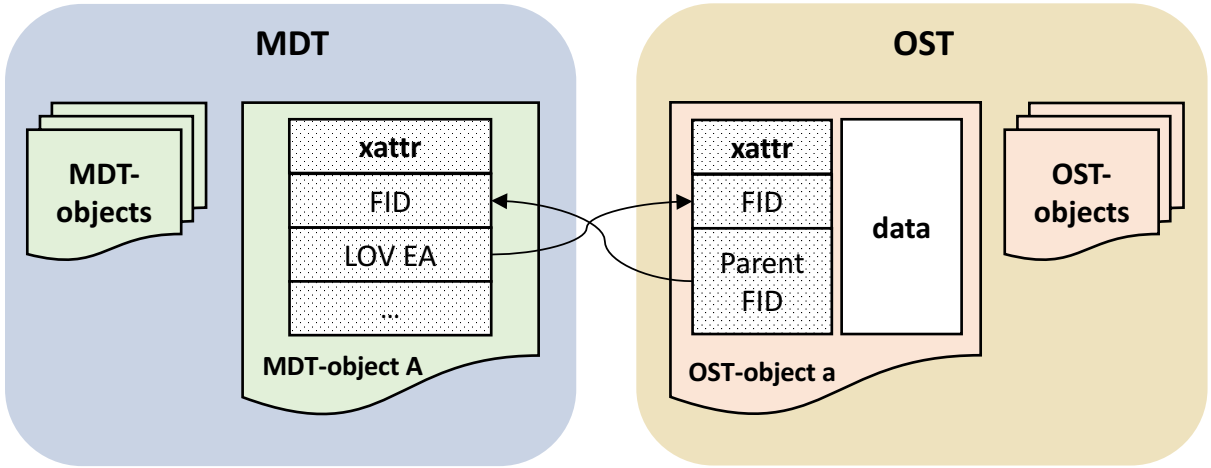
- Typical PFS architecture



- Many PFSes are designed with a *checker* component
  - e.g., LFSCK for Lustre, BeeGFS-FSCK for BeeGFS, PV2FS-FSCK for OrangeFS
  - Detect and repair inconsistencies
- FSCKs have predefined ***checker policies***

Examples of PFS Checker Policies

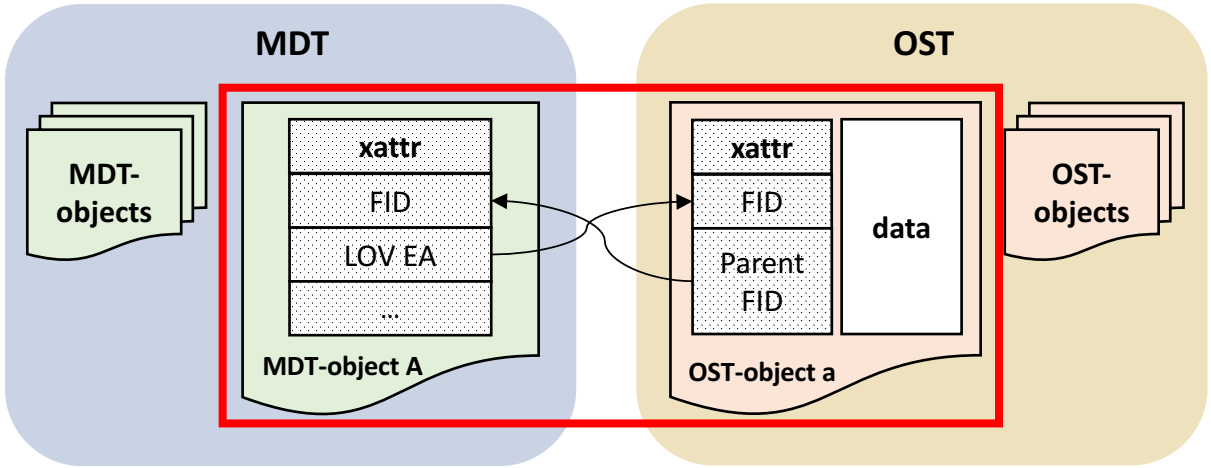
- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID
  - OST-object's Parent FID matches to MDT-object's FID



Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

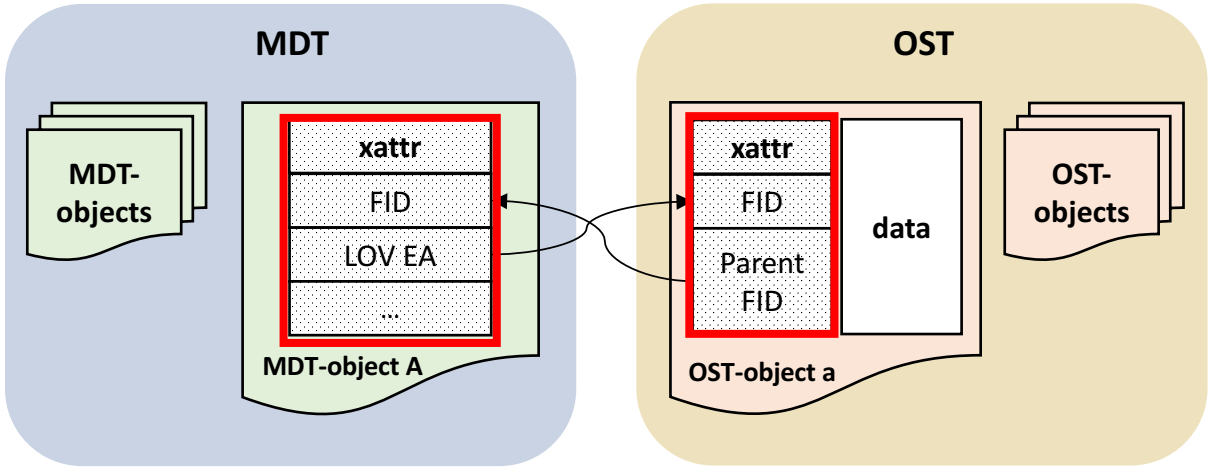
- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID
  - OST-object's Parent FID matches to MDT-object's FID



Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

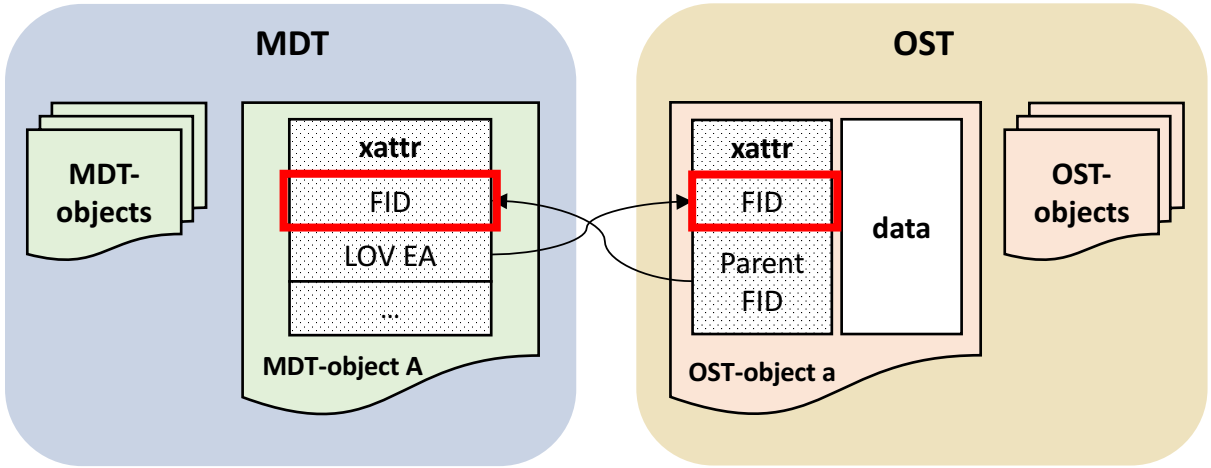
- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID
  - OST-object's Parent FID matches to MDT-object's FID



Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

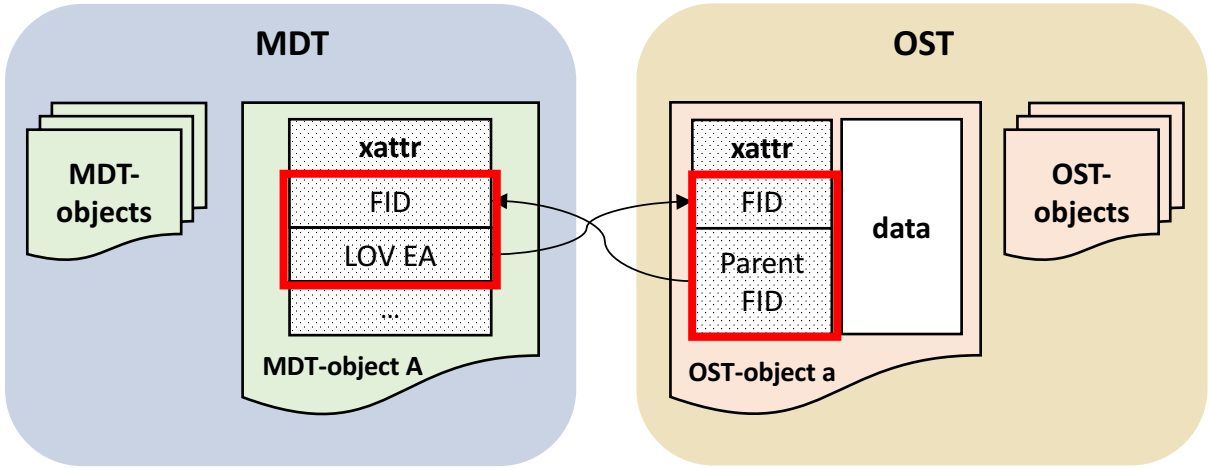
- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID
  - OST-object's Parent FID matches to MDT-object's FID



Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

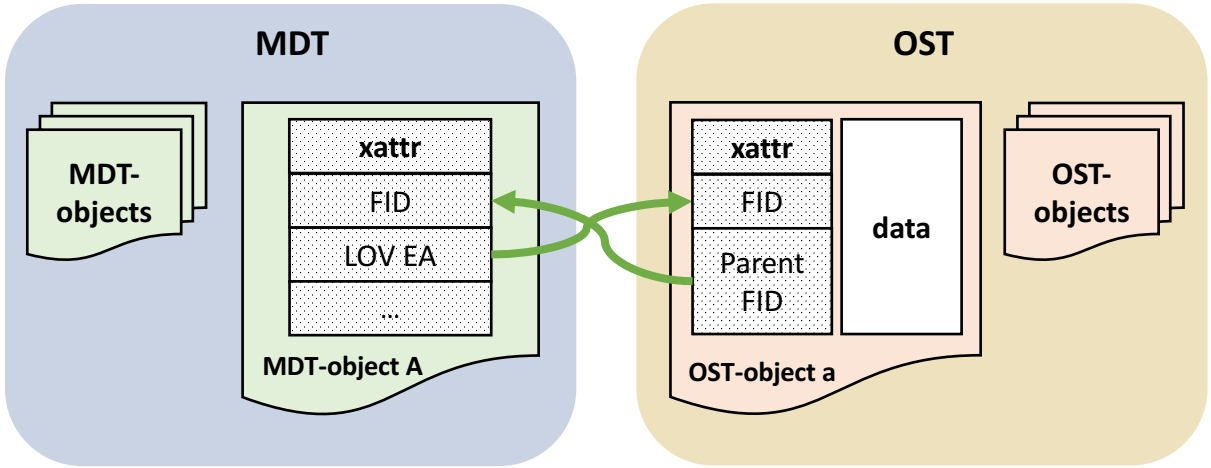
- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID
  - OST-object's Parent FID matches to MDT-object's FID



Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID
  - OST-object's Parent FID matches to MDT-object's FID

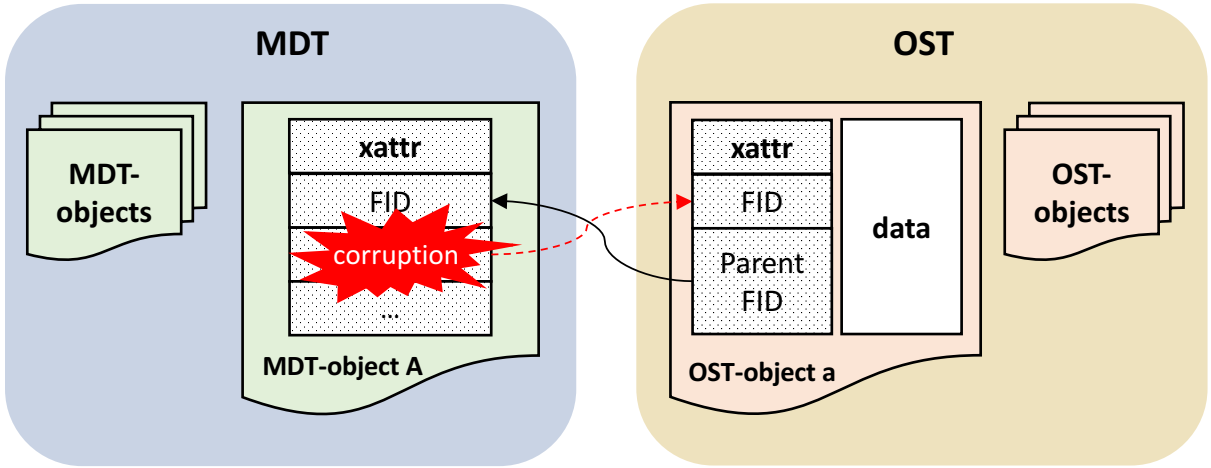


Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

- Lustre's LFSCK Policy: mapping between MDT-object and OST-object

Corruption 1
  - ~~MDT-object's LOV EA matches to OST-object's FID~~
  - OST-object's Parent FID matches to MDT-object's FID



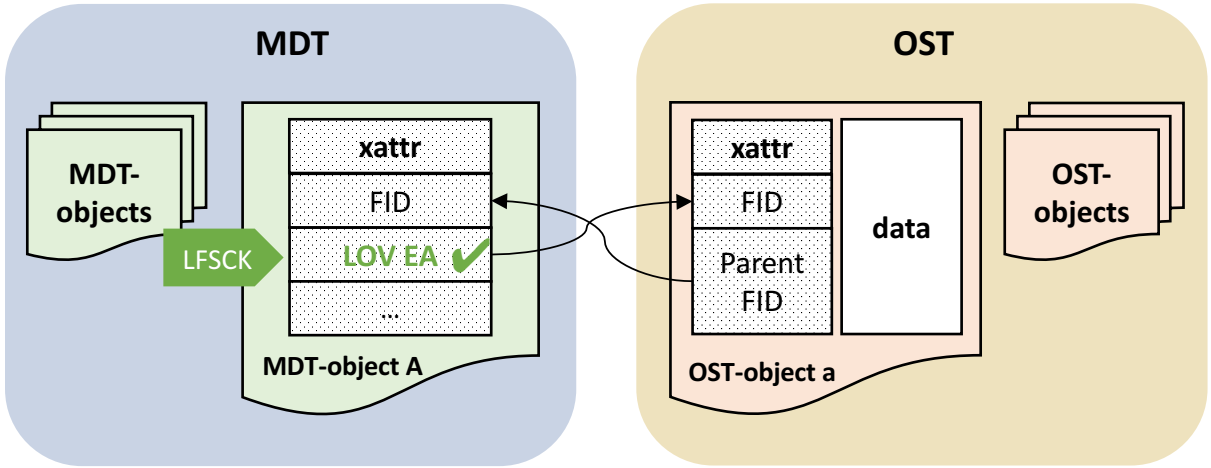
Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

- Lustre's LFSCK Policy: mapping between MDT-object and OST-object

~~Corruption 1~~ • MDT-object's LOV EA matches to OST-object's FID **Fixed!**

- OST-object's Parent FID matches to MDT-object's FID

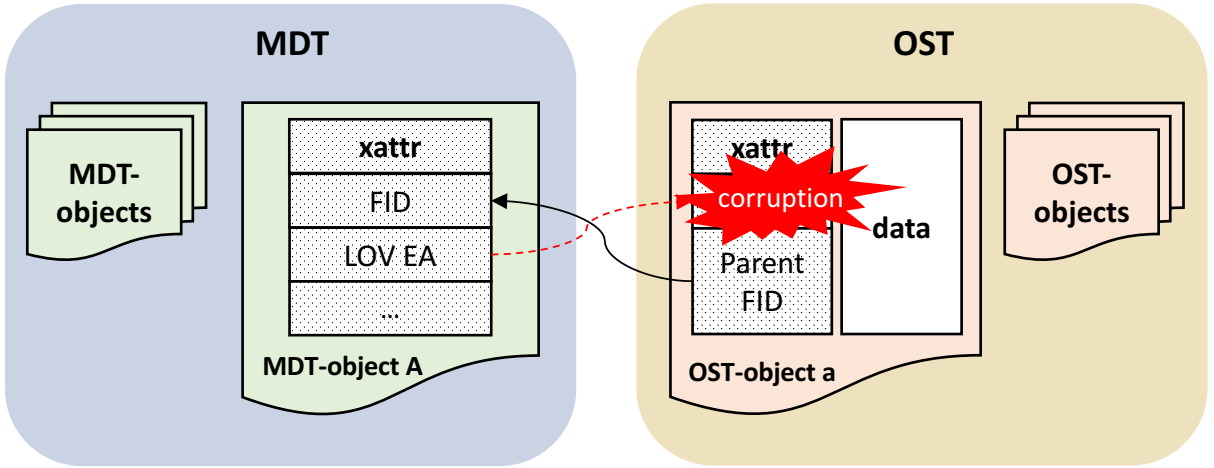


Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID
  - ~~OST-object's Parent FID matches to MDT-object's FID~~

Corruption 2

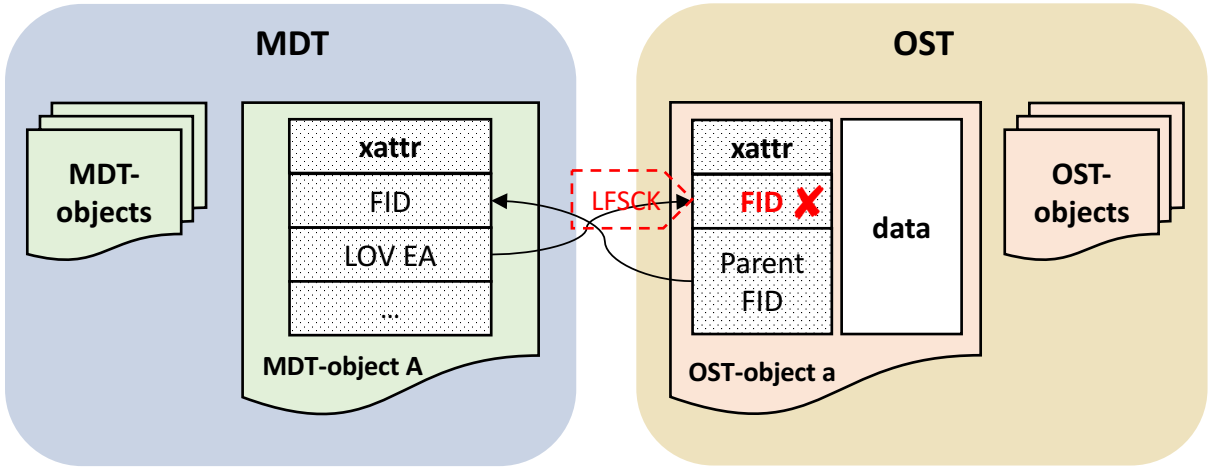


Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Examples of PFS Checker Policies

- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID

Corruption 2 ~~• OST-object's Parent FID matches to MDT-object's FID~~ **Cannot be fixed!**



Structures	Meaning
xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

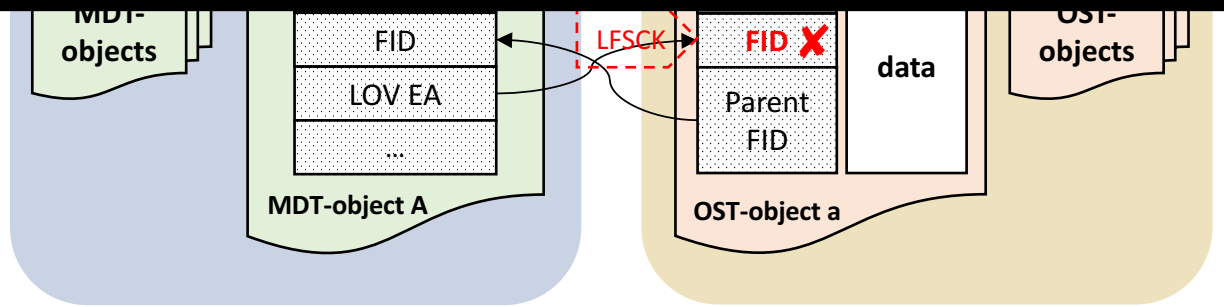
Examples of PFS Checker Policies

- Lustre's LFSCK Policy: mapping between MDT-object and OST-object
  - MDT-object's LOV EA matches to OST-object's FID
  - ~~OST-object's Parent FID matches to MDT-object's FID~~

Corruption 2

Cannot be fixed!

LFSCK's policy is incomplete!



xattr	inode extended attribute
FID	a global ID of an Lustre object
LOV EA	stores child object's FID
Parent FID	stores parent object's FID

Our Contributions

- A systematic approach to analyze PFS checker policies
  - PFS type-aware fault injection
  - PFS consistency model & taxonomy

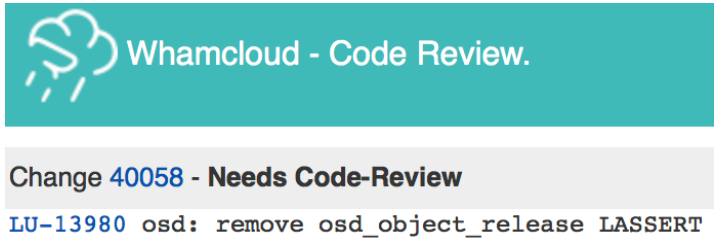
Our Contributions

- A systematic approach to analyze PFS checker policies
  - PFS type-aware fault injection
  - PFS consistency model & taxonomy
- A comprehensive study on the checkers of two widely used PFSes

Lustre®

BeeGFS®

  - Has exposed 33 suboptimal repairs
  - Has exposed 2 abnormal behaviors (e.g., kernel panic), which has led to 1 new patch on Lustre



Outline

- ~~Motivation & Contributions~~
- Methodology
  - PFS type-aware fault injection
  - Fault models
  - PFS consistency model
  - PFS checker taxonomy
- Experimental Result
- Conclusion & Future Work

PFS Type-aware Fault Injection

- Why type-aware fault injection
  - Key observation
    - PFS metadata and the local file system metadata are closely correlated
    - E.g., Both Lustre and BeeGFS has metadata structures stored in inode extended attribute (xattr)

PFS name	metadata structures in xattr	shared metadata structures with Ext4 inode
Lustre	FID, LOV EA, parent FID, linkEA	nlink
BeeGFS	fhgfs	nlink, size

  - Benefits of fine-grained fault injection
    - reveal PFS checker policies precisely
    - Enable analyzing the contract between PFS checker and local FS

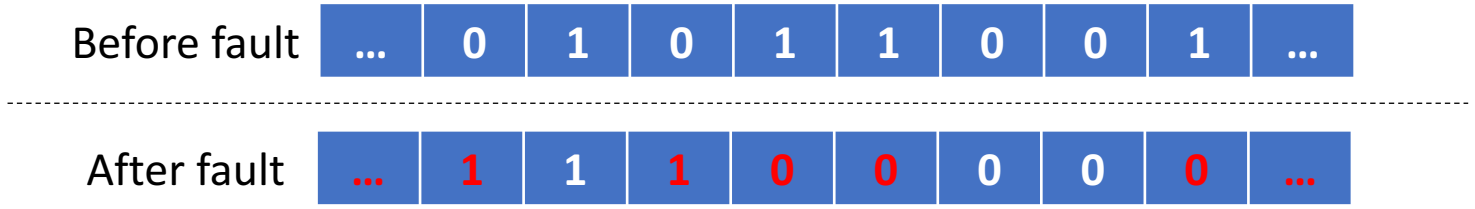
Fault Models

- Four fault models to capture the typical corruptions that may occur in the local storage stack and be exposed to the PFS checker
  - *junk, out-of-sync, zero, duplicate*

Fault Models

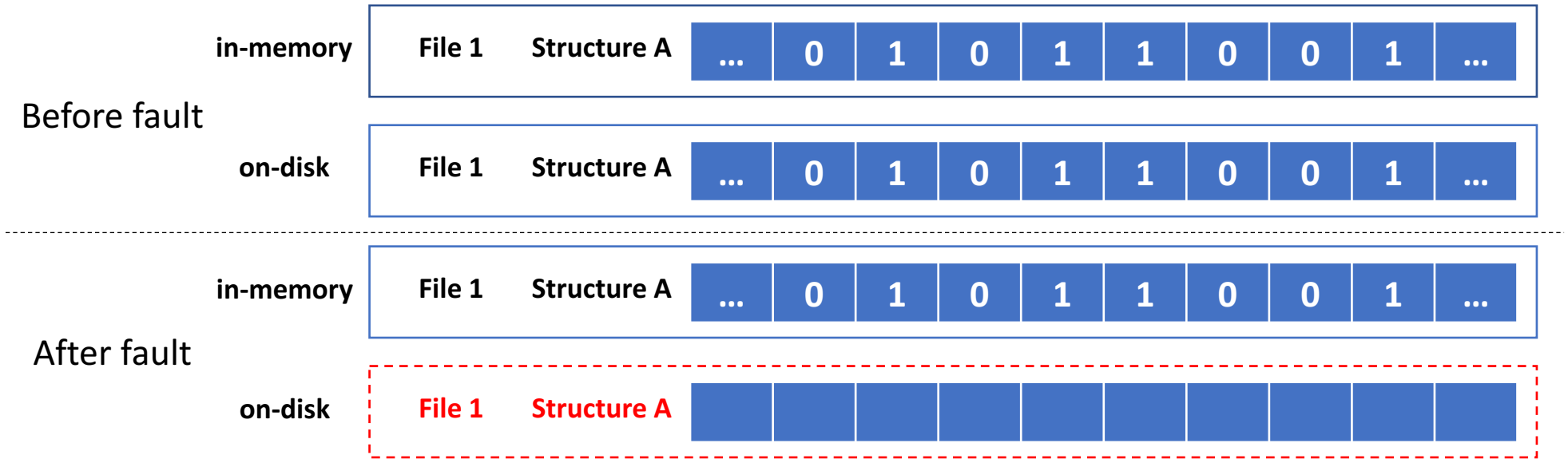
- Four fault models to capture the typical corruptions that may occur in the local storage stack and be exposed to the PFS checker
  - *junk, out-of-sync, zero, duplicate*

- Fault model #1: *junk*
  - Bytes of the on-disk structure are replaced by random values
    - Caused by disk corruptions, local FS bugs, etc



Fault Models

- Fault model #2: *out-of-sync*
  - In-memory copy of the structure is inconsistent with on-disk copy
    - Caused by software bugs ,memory/disk corruptions, etc



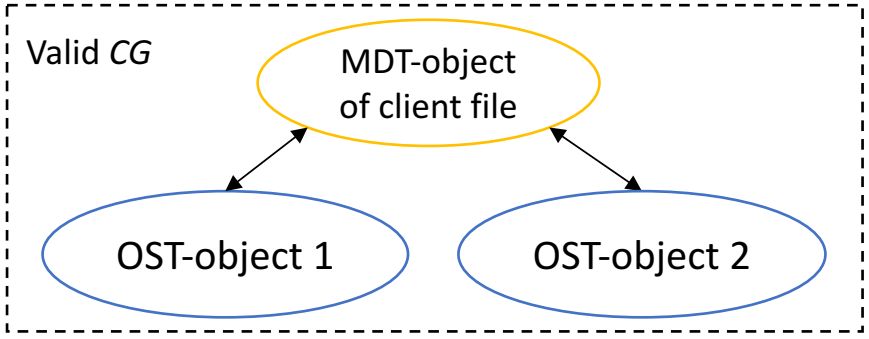
- Please refer to our paper for fault models #3 & #4

PFS Consistency Model

- General principles that PFS checkers should ensure to maintain PFS integrity
  - Applicable to diverse PFSes
  - Include the definition of Consistency Group & 6 consistency rules

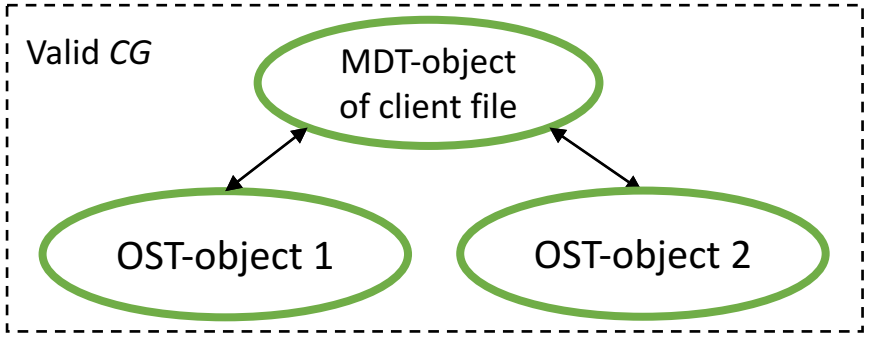
PFS Consistency Model

- General principles that PFS checkers should ensure to maintain PFS integrity
  - Applicable to diverse PFSes
  - Include the definition of Consistency Group & 6 consistency rules
- *Consistency Group (CG)*
  - Include an MDT-object and all its associated child OST-objects



PFS Consistency Model

- General principles that PFS checkers should ensure to maintain PFS integrity
  - Applicable to diverse PFSes
  - Include the definition of Consistency Group & 6 consistency rules
- *Consistency Group (CG)*
  - Include an MDT-object and all its associated child OST-objects
  - Consistency rules
    - *CG-rule1*: every object in a CG should be consistent individually

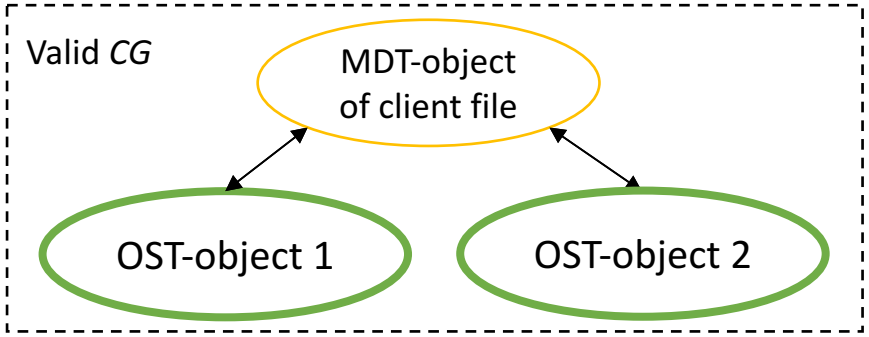


PFS Consistency Model

- General principles that PFS checkers should ensure to maintain PFS integrity
  - Applicable to diverse PFSes
  - Include the definition of Consistency Group & 6 consistency rules
- *Consistency Group (CG)*
  - Include an MDT-object and all its associated child OST-objects
  - Consistency rules
    - *CG-rule2*: one MDT-object of a client directory maps to no child OST-object

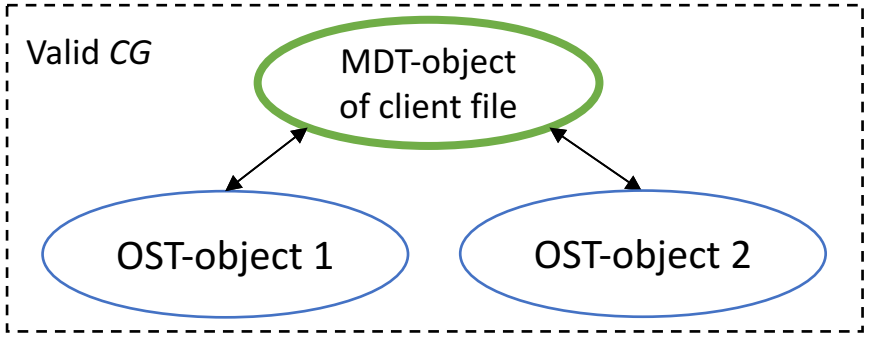
PFS Consistency Model

- General principles that PFS checkers should ensure to maintain PFS integrity
  - Applicable to diverse PFSes
  - Include the definition of Consistency Group & 6 consistency rules
- *Consistency Group (CG)*
  - Include an MDT-object and all its associated child OST-objects
  - Consistency rules
    - *CG-rule3*: one MDT-object of a client file maps to at least one child OST-object



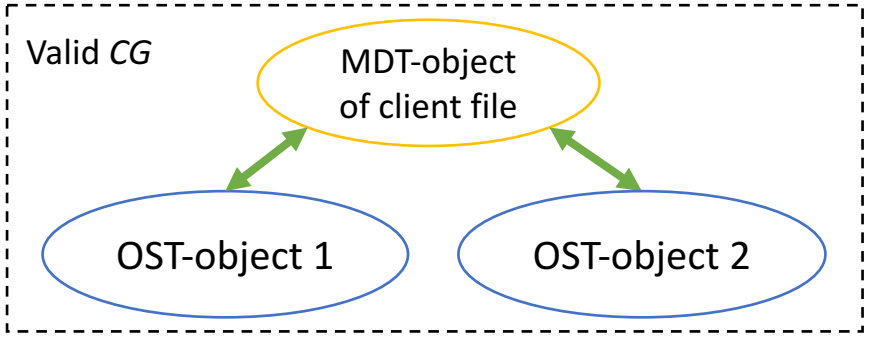
PFS Consistency Model

- General principles that PFS checkers should ensure to maintain PFS integrity
  - Applicable to diverse PFSes
  - Include the definition of Consistency Group & 6 consistency rules
- *Consistency Group (CG)*
  - Include an MDT-object and all its associated child OST-objects
  - Consistency rules
    - *CG-rule4*: one OST-object maps to one and only one parent MDT-object



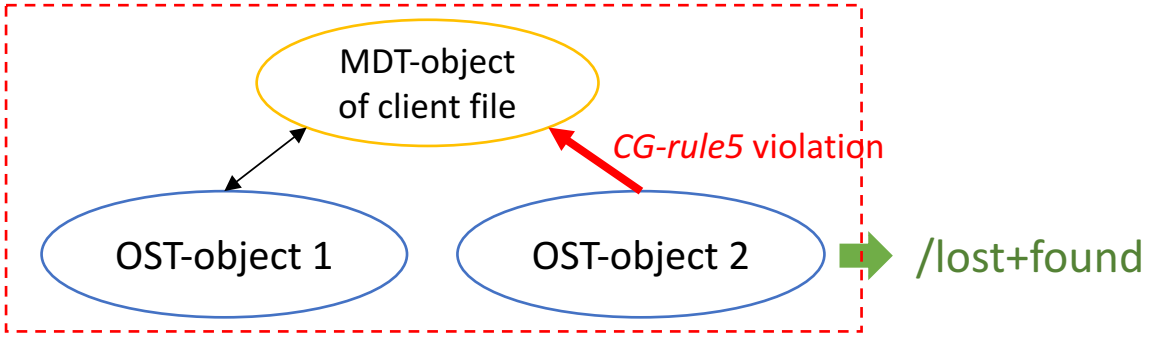
PFS Consistency Model

- General principles that PFS checkers should ensure to maintain PFS integrity
  - Applicable to diverse PFSes
  - Include the definition of Consistency Group & 6 consistency rules
- *Consistency Group (CG)*
  - Include an MDT-object and all its associated child OST-objects
  - Consistency rule
    - *CG-rule5*: the mapping b/w a parent MDT-object and a child OST-object is bidirectional



PFS Consistency Model

- General principles that PFS checkers should ensure to maintain PFS integrity
  - Applicable to diverse PFSes
  - Include the definition of Consistency Group & 6 consistency rules
- *Consistency Group (CG)*
  - Include an MDT-object and all its associated child OST-objects
  - Consistency rules
    - *CG-rule6*: an object violating previous rules may only exist in a specified location



PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results
D_{zero}	PFS checker finishes normally but misses all CG corruptions
$D_{par.}$	PFS checker partially detects CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely

Repair levels	Definition
$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
R_{zero}	PFS checker reports failure on repair
$R_{par.}$	PFS checker partially fixes CG corruptions
$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results
D_{zero}	PFS checker finishes normally but misses all CG corruptions
$D_{par.}$	PFS checker partially detects CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely

Repair levels	Definition
$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
R_{zero}	PFS checker reports failure on repair
$R_{par.}$	PFS checker partially fixes CG corruptions
$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results
D_{zero}	PFS checker finishes normally but misses all CG corruptions
$D_{par.}$	PFS checker partially detects CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely

Repair levels	Definition
$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
R_{zero}	PFS checker reports failure on repair
$R_{par.}$	PFS checker partially fixes CG corruptions
$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results
D_{zero}	PFS checker finishes normally but misses all CG corruptions
$D_{par.}$	PFS checker partially detects CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely

Repair levels	Definition
$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
R_{zero}	PFS checker reports failure on repair
$R_{par.}$	PFS checker partially fixes CG corruptions
$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results
D_{zero}	PFS checker finishes normally but misses all CG corruptions
$D_{par.}$	PFS checker partially detects CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely

Repair levels	Definition
$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
R_{zero}	PFS checker reports failure on repair
$R_{par.}$	PFS checker partially fixes CG corruptions
$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results
D_{zero}	PFS checker finishes normally but misses all CG corruptions
$D_{par.}$	PFS checker partially detects CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely

Repair levels	Definition
$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
R_{zero}	PFS checker reports failure on repair
$R_{par.}$	PFS checker partially fixes CG corruptions
$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results
D_{zero}	PFS checker finishes normally but misses all CG corruptions
$D_{par.}$	PFS checker partially detects CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely

Repair levels	Definition
$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
R_{zero}	PFS checker reports failure on repair
$R_{par.}$	PFS checker partially fixes CG corruptions
$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results
D_{zero}	PFS checker finishes normally but misses all CG corruptions
$D_{par.}$	PFS checker partially detects CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely

Repair levels	Definition
$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
R_{zero}	PFS checker reports failure on repair
$R_{par.}$	PFS checker partially fixes CG corruptions
$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

PFS Checker Taxonomy

- A general characterization of checker policies
  - Qualitatively measures the policies
  - Enable cross-PFS comparison
- Include 4 **D**etection levels & 4 **R**epair levels based on consistency model

Detection levels	Definition	Repair levels	Definition
$D_{abn.}$	PFS checker behaves abnormally w/o reporting detection results	$R_{wro.}$	PFS checker fixes CG corruptions in a wrong way
D_{zero}	PFS checker finishes normally but misses all CG corruptions	R_{zero}	PFS checker reports failure on repair
$D_{par.}$	PFS checker partially detects CG corruptions	$R_{par.}$	PFS checker partially fixes CG corruptions
$D_{com.}$	PFS checker detects CG corruptions completely	$R_{com.}$	PFS checker fixes corruptions and CGs're valid again

Outline

- ~~Motivation & Contributions~~
- ~~Methodology~~
  - ~~PFS type-aware fault injection~~
  - ~~Fault models~~
  - ~~PFS consistency model~~
  - ~~PFS checker taxonomy~~
- Experimental Results
- Conclusion & Future Work

Experimental Results

- Studied 11 Lustre structures and 7 BeeGFS structures

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Experimental Results

- Studied 11 Lustre structures and 7 BeeGFS structures

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Experimental Results

- Studied 11 Lustre structures and 7 BeeGFS structures

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Experimental Results

- Studied 11 Lustre structures and 7 BeeGFS structures

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Experimental Results

- Studied 11 Lustre structures and 7 BeeGFS structures

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	Rzero	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Experimental Results

- 14 cases: checkers repair CG corruptions completely

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Experimental Results

- 18 cases: checkers detects CG corruptions but can't repair completely

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Experimental Results

- 12 cases: checkers only check the in-memory copy of the structure
  - Could potentially miss corruptions of on-disk structures

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Experimental Results

- 2 cases: LFSCK triggers kernel panic
  - Has been confirmed by developers and led to 1 new patch
    - WhamCloud Community Jira: LU-13980, 09/26/2020

Lustre Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
MDT-object	—		—		—		$D_{abn.}$	R_{zero}
OST-object	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{abn.}$	R_{zero}
llog record	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
FID on MDT	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$
FID on OST	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	$D_{com.}$	R_{zero}	D_{zero}	R_{zero}
FLDB	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		D_{zero}	R_{zero}
OI table	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		$*D_{com.}$	$R_{com.}$
LOV EA	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$	$*D_{com.}$	$R_{com.}$
PFID	$D_{par.}$	$R_{par.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	D_{zero}	R_{zero}
linkEA	$D_{com.}$	$R_{com.}$	$D_{com.}$	$R_{com.}$	$D_{com.}$	$+R_{com.}$	$D_{com.}$	$R_{com.}$
nlink	D_{zero}	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

BeeGFS Structures	*junk*		*zero*		*duplicate*		*out-of-sync*	
dentry-by-name (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
dentry-by-ID (MDT-object)	—		—		—		$D_{com.}$	$R_{com.}$
chunk (OST-object)	D_{zero}	R_{zero}	D_{zero}	R_{zero}	—		$D_{par.}$	R_{zero}
fghfs	$D_{com.}$	$R_{wro.}$	$D_{com.}$	$R_{wro.}$	$D_{com.}$	R_{zero}	$D_{com.}$	$R_{wro.}$
content directory	—		—		—		$D_{com.}$	$R_{com.}$
nlink	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	
size	$*D_{com.}$	R_{zero}	$*D_{com.}$	R_{zero}	—		—	

Outline

- ~~Motivation & Contributions~~
- ~~Methodology~~
  - ~~PFS type-aware fault injection~~
  - ~~Fault models~~
  - ~~PFS consistency model~~
  - ~~PFS checker taxonomy~~
- ~~Experimental Result~~
- Conclusion & Future Work

Conclusion & Future Work

- A systematic approach to study PFS checkers
  - Has led to a new patch on Lustre

- Future work
  - More automation (e.g., apply fuzzing techniques)
  - Study other PFSes (e.g., OrangeFS, Ceph)
  - Improve PFS checkers
    - Policy completeness
    - Performance

Conclusion & Future Work

- A systematic approach to study PFS checkers
  - Has led to a new patch on Lustre

- Future work
  - More automation (e.g., apply fuzzing techniques)
  - Study other PFSes (e.g., OrangeFS, Ceph)
  - Improve PFS checkers
    - Policy completeness
    - Performance

Thank you & Questions?